Systems Manual - Social Hour

This systems manual instructs new programmers on how to understand the code written for the project. Each function and data element is organized by basic type.

User Credentials and Logging In

Firebase Authentication is handled by the Login activity, handled by the "Login" activity at the main function. Login is the entry point for users entering the application, with layout specified in the layout XML directory. Firebase authentication is handled specifically by the FirebaseAuthWithGoogle function, which is triggered by the SignInButton "signInButton" on the main screen. If the user has already authenticated with firebase, the user will not be presented with the login screen; rather the user will go straight into the main application.

frontend_activity

This is where most of the work happens. Nearly all of the databases are managed here, as well as a ViewPager which handles pager fragments. Two floating action buttons appear on the main screen, and are persistent throughout the entirety of the activity: fab, and fabcal. In addition, the FrontendActivity also handles a menubar that allows the user to enter the settings activity.

frontend_activity: Database Management

All of the databases in the application are handled by DatabaseReferences, which are references to Google Firebase.

- private_user_database: handles all of the data that is only meant to be accessed locally: contains data that only the user can see for themselves, stored in a PrivateUserData object.
- public_user_database: handles all of the data that is meant to be accessed by any user of the application: entries in public_user_database are meant to be handled by many different users, stored in PublicUserData class
- public_event_database: the event repository for the whole application; filtered by relevant events (whether or not the user actually needs to pull the event in for viewing), stored in EventItem class
- friend_connections_database: handles friend connections within firebase, stored in FriendItem class
- group_database: handles groups within the database, stored in GroupItem class

frontend_activity: dashboard

This is where users can see a feed of new events popping up throughout the course of the application. This is handled as a fragment of frontend_activity, and contains a RecyclerView that displays all of the items in EventData. In addition, the dashboard contains a function that updates the display, which is frequently triggered by frontend_activity.

frontend_activity: friends_menu

This is where users can see a feed of new groups popping up throughout the course of the application. This is handled as a fragment of frontend_activity, and contains a RecyclerView that displays all of the items in FriendData. In addition, the dashboard contains a function that updates the display, which is frequently triggered by frontend_activity.

frontend_activity: groups_menu
This is where users can see a feed of new friend connections popping up throughout the course of the application. This is handled as a fragment of frontend_activity, and contains a RecyclerView that displays all of the items in GroupData. In addition, the dashboard contains a function that updates the display, which is frequently triggered by frontend_activity.

AppCompatPreferenceLibrary
Functions supplied by Google so that we don't have to worry about supporting older devices. Irrelevant to the programmer, these functions only need to be looked at when something breaks horribly. Since the code is supplied by Google, there will likely be no such errors regarding this, so no need to discuss more.

calendar_activity:
The calendar activity uses a date picker to allow the user to filter all of the events in the database that are relevant to them based on the date. The date picker has an OnDateChanged listener that updates the RecyclerView whenever an individual clicks an item. The updating of the display is handled by the notifyDataSetChanged function.

add_friend_activity
Adding friends is handled by add_friend_activity. This activity is the only one besides frontend_activity that contains a database listener, and also has a search bar that allows users to search for events. add_friend_activity's data is stored under public_user_search_adapter, which then has a filter that allows users to actually modify the data to be stored.

settings_activity
User preference management is handled by the settings activity. The settings activity is accessed through the menu bar of the frontend_activity. All of the current settings of the user are passed through an intent, and used to populate the fields in the settings. Here, a user is able to change their display name, modify 12 or 24 hour time format, and log out of the application. After all of this is finished, all of the data is passed out through an intent again, and looked at back in the main activity.

Data Structures:
EventItem: handles all of the data that is stored in individual events. This includes the author, whether or not the event takes place all day, whether or not the event is a part of a group, the start date, the end date, and the creation date.

GroupItem: handles all of the data that are stored in individual groups.
EventData: handles an ArrayList of events maintained throughout the application. This is populated through the frontend_activity through the maintenance of the group database and the public database, and is accessed in the dashboard activity and calendar activity.

GroupData: handles an ArrayList of GroupItems, which are all of the groups that are relevant to the user. This is populated thorugh the frontend_activity through the maintenance of the group database, and is accessed in the groups_menu activity and the Add_Event_Activity.

FriendData: handles an ArrayList of FriendItems, which are all of the friend requests that are relevant to the user. This is populated through the frontend_activity through the maintenance of the friend database, and is accessed in the friends_menu activity and the add_friend_activity.

Layout and resource files are nested in their own folders under resources.

Any other manual needs are taken care of by examining documentation for Android Studio, Google Firebase, and other necessary documentation required to effectively interface with those systems.